Ahmad Farhan H A
1506732766
APAP - A

1. Method yang diubah pada Latihan Merubah **SelectAllStudents**

```
    <h3>Kuliah yang diambil</h3>
    <ul th:each="course,iterationrStatus: ${student.courses}">
        <li th:text="${course.name} + '-' + ${course.credits} + ' sks'">
            Nama kuliah-X SKS
        </li>
    </ul>
<!--
    <a th:href="'/student/delete/' + ${student.npm}" >Delete Data </a> <br/>
    <a th:href="'/student/update/' + ${student.npm}" >Update Data </a> <br/>
 -->
    <hr/>
```

Pada html viewall saya menambahkan “Kuliah yang diambil” beserta kode yang akan menampilkan nama mata kuliah beserta jumlah sks yang diambil oleh student. Selain itu saya memblok kode yang memungkinkan menghapus dan mengupdate data student.

```
@Select("select npm, name, gpa from student")
@Results(value = {
        @Result(property="npm", column="npm"),
        @Result(property="name", column="name"),
        @Result(property="gpa", column="gpa"),
        @Result(property="courses", column="npm",
                javaType = List.class,
                many=@Many(select="selectCourses"))
})
List<StudentModel> selectAllStudents ();
```

Pada studentMapper saya mengubah method selectAllStudent sehingga method ini akan mengambil npm, nama, dan gpa dari database dan mengembalikan list berupa StudentModel. Property menunjukkan nama variable pada class studentModel dan column menunjukkan hasil kolom hasil kueri di database. Kemudian untuk courses diisi menggunakan npm karena npm yang akan menjadi parameter di method selectCourses. Dan variabel many diset dengan method selectCourses yang akan mengisi variabel courses.

2. Method yang diubah pada Latihan Menambahkan **View** pada **Course**

Ahmad Farhan H A
1506732766
APAP - A

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
    <head>
        <title>View Course by ID</title>
    </head>
    <body>
        <h3 th:text="'ID = ' + ${course.idCourse}">Course ID</h3>
        <h3 th:text="'Nama = ' + ${course.name}">Course Name</h3>
        <h3 th:text="'SKS = ' + ${course.credits}">Course Credits</h3>

        <h3>Mahasiswa yang mengambil</h3>
        <ul th:each="student: ${course.students}">
            <li th:text="${student.npm} + ' - ' + ${student.name}">
                NPM - Nama
            </li>
        </ul>

    </body>
</html>
```

Pertama saya membuat html viewcourse yang akan menampilkan informasi course yang diinginkan beserta mahasiswa yang mengambil course tersebut.

```
CourseModel selectCourse (String idCourse);
```

Lalu saya menambahkan method selectCourse yang mengembalikan CourseModel dan memiliki parameter idCourse di StudentService.

```
@Override
public CourseModel selectCourse (String idCourse)
{
    log.info ("select course with id_course {}", idCourse);
    return studentMapper.selectCourse (idCourse);
}
```

Kemudian saya mengimplementasikan method selectCourse di StudentServiceDatabase.

```
@Select("select id_course, name, credits from course where id_course = #{id_course}")
@Results(value = {
        @Result(property="idCourse", column="id_course"),
        @Result(property="name", column="name"),
        @Result(property="credits", column="credits"),
        @Result(property="students", column="id_course",
                javaType = List.class,
                many=@Many(select="selectStudents"))
})
CourseModel selectCourse (@Param("id_course") String idCourse);

@Select("select student.npm, name, gpa " +
        "from studentcourse join student " +
        "on studentcourse.npm = student.npm " +
        "where studentcourse.id_course = #{id_course}")
List<StudentModel> selectStudents (@Param("id_course") String idCourse);
```

Lalu saya membuat method selectCourse dan selectStudents pada StudentMapper. Method selectStudents mengembalikan mengembalikan data student yang mengambil course sesuai parameter idCourse. Sedangkan method selectCourse mengembalikan id, nama, jumlah sks, serta nama-nama mahasiswa yang mengambil course berdasarkan parameter idCourse.

```
@RequestMapping("/course/view/{id}")
public String viewCoursePath(Model model,
        @PathVariable(value = "id") String idCourse)
{
    CourseModel course = studentDAO.selectCourse (idCourse);

    if (course != null) {
        model.addAttribute ("course", course);
        return "viewcourse";
    } else {
        model.addAttribute ("id_course", idCourse);
        return "course-not-found";
    }
}
```

Terakhir saya membuat method viewCoursePath di StudentController. Method ini mengambil course sesuai dengan parameter idCourse. Jika terdapat idCourse di database, maka akan berpindah ke halaman viewcourse. Sedangkan apabila tidak ada data idCourse di database, maka akan berpindah ke halaman course-not-found.